Reflective Report

In completing the project, I split the work into four parts:

STEP 1: Reading and Parsing the input files

STEP 2: Querying the data for the specified information

STEP 3: Search Algorithm

STEP 4: Writing to an output file

STEP 1: Reading and Parsing the input files

In reading the text files, the data structure used to store the CSV data was a hashpMap. A HashMap was used for its ability to store key-value pairs allowing me to format the CSV in a way that allowed for easy access to specific information. In general, they stored a String Key and List of objects representing the airport, airline or route. The object's data are read line by line from the CSV file and split with various parts passed in as the airport, airline, or route parameter to be instantiated. The text file storing the start and destination is read into a list.

STEP 2: Querying the data for the specified information

In this step, the data is queried, and the appropriate information is returned based on the given start and destination cities. Airports are chosen to search for any flights connecting those airports. After the search is complete, a query is repeated to ensure a valid flight path is taken.

STEP 3: Search Algorithm

Here the airport data was taken, specifically the name and id and were used to construct the node of a graph. The graph is used as a directed graph represented by an adjacency matrix. The initial intent was to use Dijkstra's shortest path algorithm to find the best optimal path. However, I ran into errors as some of the airport data were inconsistent. The alternative used was breadth-first search, iteratively checking all nearest nodes and returning a path List. Since BFS returns optimal for a non-decreasing function, I did not consider the weight on the edges (distance calculated from longitude and latitude).

STEP 4: Writing to the output file

The data from the BFS is then written into an output file, and the output file with the specified name is generated.

Improvement and Recommendation.

One very important lessoned I have to learn is memory management and code optimization to reduce runtime as at multiple points in the implementation, my JVM run out of memory

A few improvements that could be made to the code, I think, would be to recycle some results from running the queries instead of writing a new one each time, making the code more compact by removing unnecessary assignments and using a search algorithm that incorporates the distance between airports into the calculation of the shortest path.

For future works I will look into how to engineer the Dijkstra's algorithm to better suit the needs of the project.